Processes and Services

Unix System Admin DeCal

Christopher Cooper <cooperc>
(based on F19 lecture by keur)

Part 1: Processes

What is a process?

A process is a single instance of a program.

Processes are isolated from one another and have their own memory, threads, etc. (Additional isolation, such as filesystem or network isolation, is also possible.)

What is a process?

- PID: Process ID
- PPID: Parent’s PID
- UID: User running the process
- The program (executable) that the process is running
- The args (command line) of the process

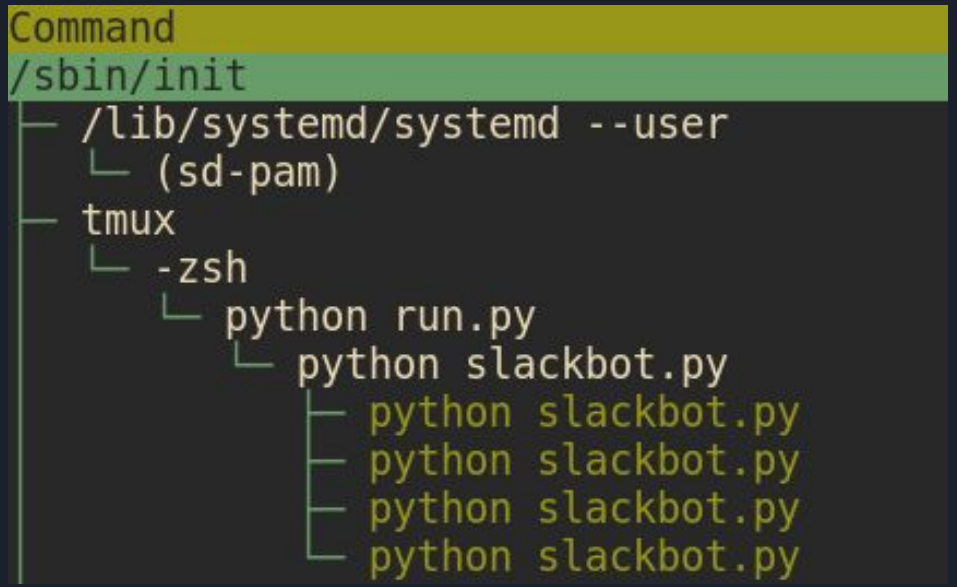
(and more…)

Process Hierarchy

Each process is created by a parent

(Except PID 1)

Processes can have many children



Processes vs Threads

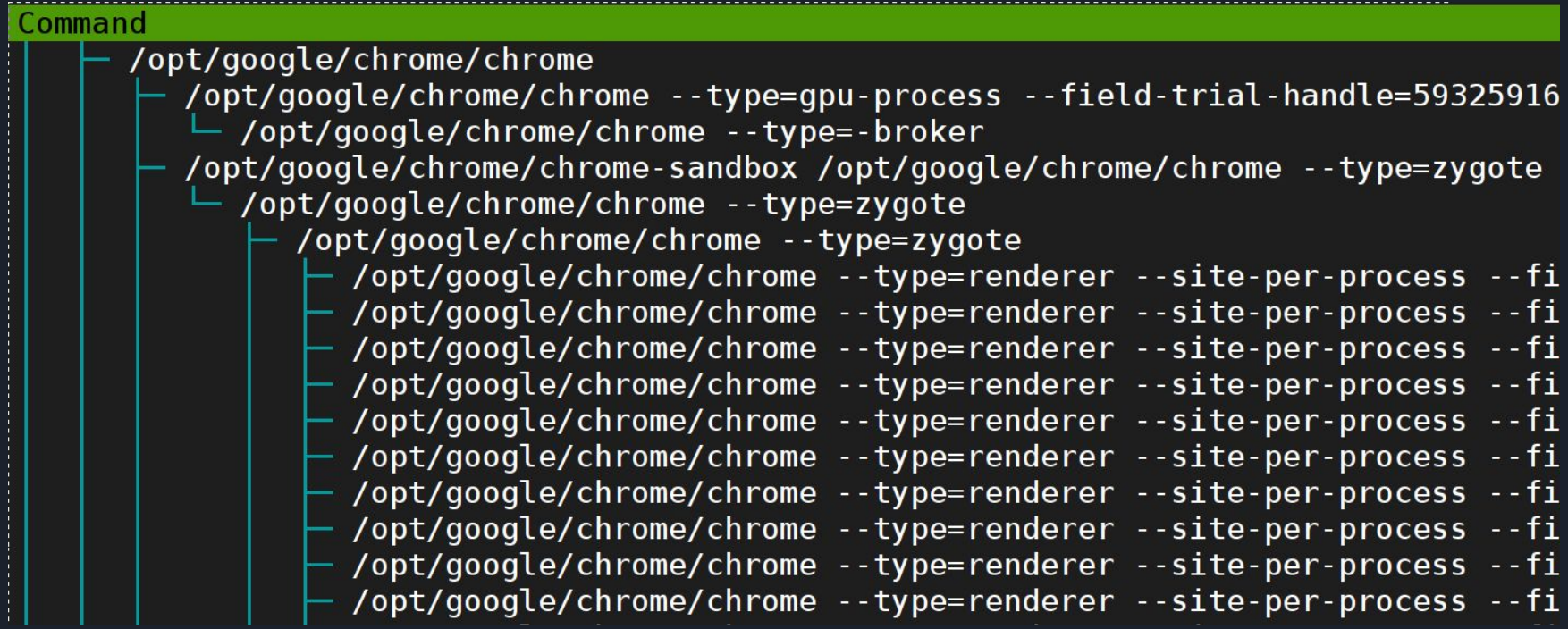
Both run code concurrently and can take advantage of parallelism

Threads are “lightweight” processes

All threads share virtual address space, code, static/globals, memory, resources (open files)

Processes must communicate over some interface (IPC)

Aside: Why the hell does Chrome spawn so many processes?



How are processes created?

A process will **fork**(2) into a two new processes, which continue from the same place

The parent keeps the original PID and the child gets a new PID

Optionally, the new process (the child) **exec**(3) and begin running a new program

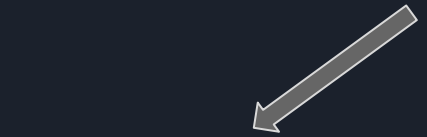
How many “henlo”s get printed?

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

How many “henlo”s get printed?

PID 1000

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```



PID 1000

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

PID 2000

```
int main( void ) {
    fork();
    printf( "henlo: %d\n", getpid() );
}
```

How many “henlo”s get printed?

```
int main( void ) {
    fork();    2^1
    fork();    2^2
    fork();    2^3
    printf( "henlo: %d\n", getpid() );
}
```

```
$ ./fork
henlo 104632
henlo 104635
henlo 104634
henlo 104633
henlo 104638
henlo 104639
henlo 104637
henlo 104636
```

Have you seen this code before?

```
:(){ :|:& };:
```

```
bomb() {
    bomb | bomb &
} bomb
```

Making the child do something

```c
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        wait( NULL );
    } else {
        /* child process */
        execv( “/bin/bash”, NULL );
    }
}
```

```
/usr/bin/zsh
 \_ ./fork_exec
     \_ [bash]
```

What does this code do?

```
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        sleep( 1 );
    } else {
        /* child process */
        exit( 1 );
    }
}
```

What does this code do?

```
\_ /usr/bin/zsh
\_ ./zombie-creator
    \_ [zombie-creator] <defunct>
```

Parent

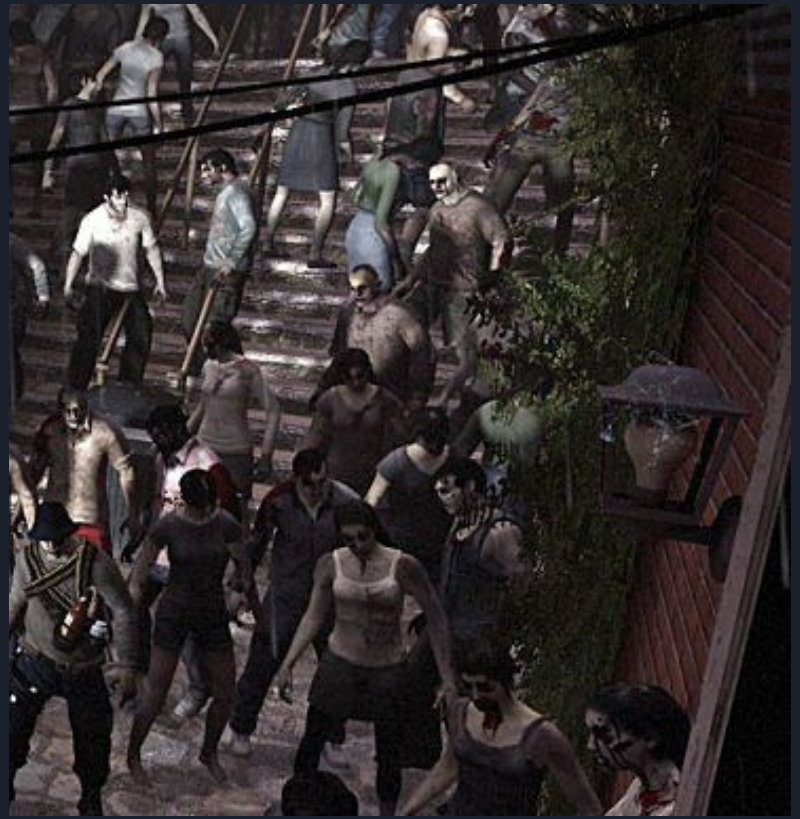
Child

Zombie Process

When a child has died but has not been “reaped”

Child metadata stays in process table so parent can collect exit status

Totally normal, all children that exit are zombies!



How to kill zombie process

124

I launched my program in the foreground (a daemon program), and then I killed it with `kill -9`, but I get a zombie remaining and I m not able to kill it with `kill -9`. How to kill a zombie process?

share edit flag

50

edited Dec 10 '14 at 18:36
octosquidopus
1,125 ● 3 ● 20 ● 38

asked Jun 5 '13 at 16:14
MOHAMED
18.2k ● 28 ● 97 ● 169

5 Answers

active oldest votes

168

A zombie is already dead, so you cannot kill it.

What happens when a process dies

```
if ( fork() > 0 ) {
    /* parent process */
    sleep( 1 );
} else {
    /* child process */
    exit( 1 );
}
```

When a process exits, returns int (exit code 0-255)

0 is success and anything else indicates an error

What happens when a process dies

```
if ( fork() > 0 ) {
    /* parent process */
    sleep( 1 );
} else {
    /* child process */
    exit( 123 );
}
```

When a process exits, returns int (exit code 0-255)

0 is success and anything else indicates an error

Parents need to wait() on children

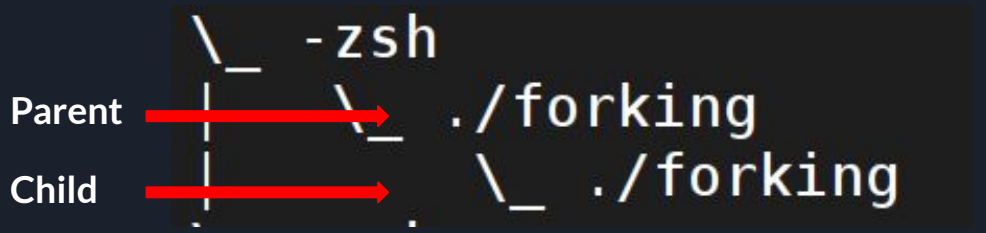
```c
int main( void ) {
    if ( fork() > 0 ) {
        int status;
        sleep( 1 );
        wait( &status );
        printf( "%d\n", WEXITSTATUS ( status ) );
    } else {
        exit( 123 );
    }
```

```
# keur @ magneto in ~/repos/temp
$ ./good-parent
123
```

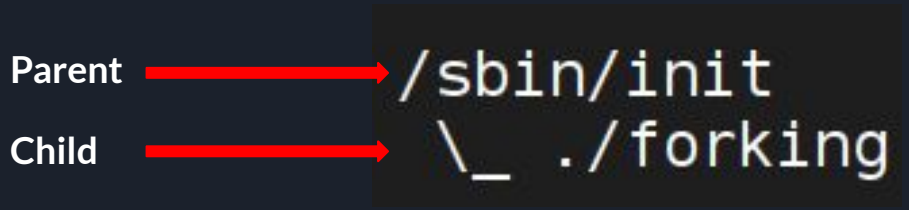
What happens if the parent exits first?

```c
int main( void ) {
    if ( fork() > 0 ) {
        /* parent process */
        sleep ( 1 );
        exit( 0 );
    } else {
        /* child process */
        sleep( 100 );
        exit( 123 );
    }
```

Process tree after start



Process tree after 1 second



What happens if the parent exits first?

If parent exits first, orphan processes are re-parented by the init process

init reaps all orphans that are zombies

When can zombies become a problem?

Parent doesn’t wait() on children

Parent is long running process

Zombie child processes never become orphans

Resource leakage!

Zombie leakage in the wild

```
kitty
 \_ /usr/bin/zsh
     \_ aerc
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
```

```
310bec27024579e7ada35585b3190ab875540804 — Kevin Kuehler  be4ea0d  master
widgets/msgview: Reap the filter command

The filter command shells out and returns almost immediately. Call
Wait() so the filter process gets reaped. Prior to this patch, aerc
creates a zombie process for every email that is viewed.

Signed-off-by: Kevin Kuehler <keur@xcf.berkeley.edu>
```

```
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
         \_ less -R
         \_ [awk] <defunct>
```

2020 update

```
12:33 <cooperc> but what about aerc????
12:35 <keur> no
12:35 <keur> 👎
13:37 <keur> i can distinctly remember why
13:37 <keur> 2019-11-21 11:33:10 keur i just found more zombies
13:37 <keur> 2019-11-21 11:34:39 ddevault you shouldn't talk about your mom that way
13:37 <keur> goodbye aerc
```

Inter-process Communication

Various ways in which processes can communicate

- Exit codes
- Signals (e.g. SIGTERM, SIGKILL, SIGINT)
- Pipes (STDIN, STDOUT, STDERR)
- Sockets (UNIX socket, IP socket)
- Message Bus (e.g. dbus on Linux)
- … and many many more...

Process Signals

- **SIGTERM**: tell a process to exit now
- **SIGKILL**: terminate process immediately
- **SIGINT**: interrupt, when you press Ctrl+C
- **SIGHUP**: the user closes the terminal window
- **SIGWINCH**: terminal window resized
- **SIGSTOP** / **SIGCONT**: stop/resume

SIGKILL and SIGSTOP cannot be handled

Working with processes: ps

ps aux

```
# keur @ magneto in ~ [12:12:32]
$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 187592 10532 ?        Ss   Oct15   0:15 /sbin/init
root           2  0.0  0.0      0     0 ?        S    Oct15   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Oct15   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Oct15   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   Oct15   0:00 [kworker/0:0H-kblockd]
root           8  0.0  0.0      0     0 ?        I<   Oct15   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    Oct15   0:01 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I    Oct15   0:14 [rcu_preempt]
root          11  0.0  0.0      0     0 ?        S    Oct15   0:03 [rcuc/0]
root          12  0.0  0.0      0     0 ?        S    Oct15   0:00 [rcub/0]
root          13  0.0  0.0      0     0 ?        S    Oct15   0:00 [migration/0]
root          14  0.0  0.0      0     0 ?        S    Oct15   0:00 [idle_inject/0]
root          16  0.0  0.0      0     0 ?        S    Oct15   0:00 [cpuhp/0]
root          17  0.0  0.0      0     0 ?        S    Oct15   0:00 [cpuhp/1]
root          18  0.0  0.0      0     0 ?        S    Oct15   0:00 [idle_inject/1]
root          19  0.0  0.0      0     0 ?        S    Oct15   0:00 [migration/1]
root          20  0.0  0.0      0     0 ?        S    Oct15   0:02 [rcuc/1]
root          21  0.0  0.0      0     0 ?        S    Oct15   0:01 [ksoftirqd/1]
root          23  0.0  0.0      0     0 ?        I<   Oct15   0:00 [kworker/1:0H-kblockd]
root          24  0.0  0.0      0     0 ?        S    Oct15   0:00 [cpuhp/2]
root          25  0.0  0.0      0     0 ?        S    Oct15   0:00 [idle_inject/2]
root          26  0.0  0.0      0     0 ?        S    Oct15   0:00 [migration/2]
root          27  0.0  0.0      0     0 ?        S    Oct15   0:03 [rcuc/2]
root          28  0.0  0.0      0     0 ?        S    Oct15   0:01 [ksoftirqd/2]
root          30  0.0  0.0      0     0 ?        I<   Oct15   0:00 [kworker/2:0H-kblockd]
root          31  0.0  0.0      0     0 ?        S    Oct15   0:00 [cpuhp/3]
root          32  0.0  0.0      0     0 ?        S    Oct15   0:00 [idle_inject/3]
root          33  0.0  0.0      0     0 ?        S    Oct15   0:00 [migration/3]
```

Working with processes: ps

ps fwaux

```
# keur @ magneto in ~ [12:17:28]
$ ps fwaux | tail -n 10
keur       70404  0.1  0.0  13448  9356 pts/16   Ss+  10:33   0:10  \_ /usr/bin/zsh
keur       80593  0.1  0.2 1014872 33864 ?       Sl   11:45   0:02 alacritty
keur       80597  0.0  0.0  12752  8604 pts/20   Ss   11:45   0:00  \_ /usr/bin/zsh
keur       80747  0.0  0.0  10148  4708 pts/20   S+   11:46   0:00      \_ man pstree
keur       80757  0.0  0.0   6504  2684 pts/20   S+   11:46   0:00          \_ less
root       81103  0.0  0.0 310816  3636 ?        Ssl  12:09   0:00 /usr/bin/pcscd --foreground --auto-exit
keur       81634  0.8  0.6 276708 98572 ?        Sl   12:12   0:02 kitty
keur       81638  0.7  0.0  12728  8668 pts/1    Ss   12:12   0:02  \_ /usr/bin/zsh
keur       82142  0.0  0.0  13184  3564 pts/1    R+   12:17   0:00      \_ ps fwaux
keur       82143  0.0  0.0   5332   760 pts/1    S+   12:17   0:00      \_ tail -n 10
```

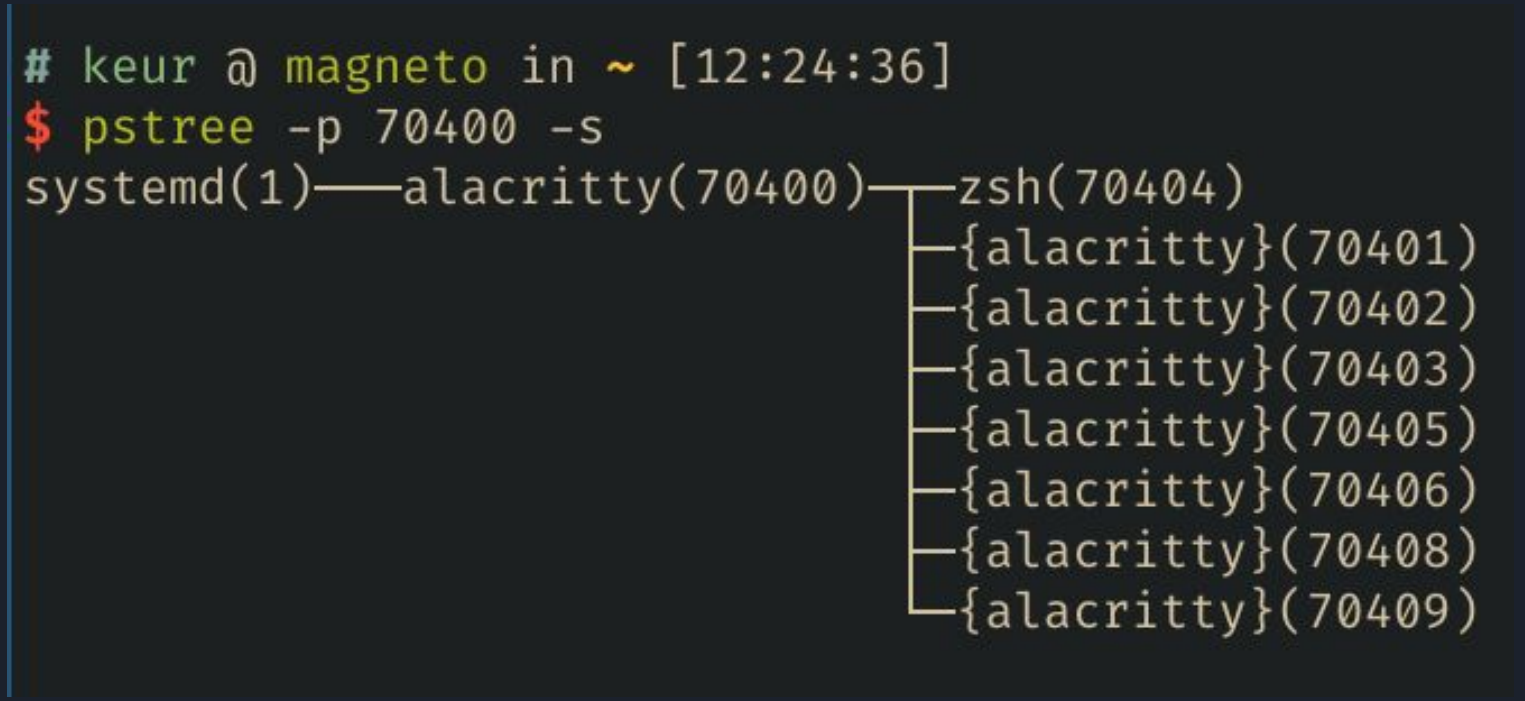
Working with processes: ps

ps -eo pid,user,cmd --sort user

```
# keur @ magneto in ~ [12:20:10]
$ ps -eo pid,user,cmd --sort user | head -n 18
    PID USER     CMD
    537 colord   /usr/lib/colord
    505 dbus     /usr/bin/dbus-daemon --system --address=systemd: --nofork
    803 keur     /usr/lib/systemd/systemd --user
    804 keur     (sd-pam)
    809 keur     /usr/bin/ssh-agent -D -a /run/user/1000/ssh-agent.socket
    817 keur     /usr/lib/gdm-wayland-session --register-session sway
    819 keur     /usr/bin/dbus-daemon --session --address=systemd: --nofork
    821 keur     sway
    829 keur     swaybg -o eDP-1 -i /home/keur/Pictures/purple.png -m fill -
l -o DP-2 -i /home/keur/Pictures/miku1.jpg -m fill
    832 keur     swaybar -b bar-0
    834 keur     mako
    837 keur     i3status
    891 keur     alacritty
    895 keur     /usr/bin/zsh
    903 keur     /usr/bin/gpg-agent --supervised
    948 keur     mosh-client -# keur@23.239.23.104 | 23.239.23.104 60015
    954 keur     scdaemon --multi-server
```

Working with processes: ps

pstree



Working with processes: kill, pkill

Send TERM to process

- `$ kill 100`

Reload SSH process

- `$ pkill -HUP sshd`

Send SIGKILL to chrome

- `$ pkill -9 google-chrome-stable`

Working with processes: pgrep

```
# keur @ magneto in ~ [13:08:13]
$ pgrep brave
69169
69172
69174
69195
69201
69370
69371
69376
69412
69434
69439
69446
69609
70623
70703
72760
74619
```

Good for command chaining with xargs

Working with processes: pgrep

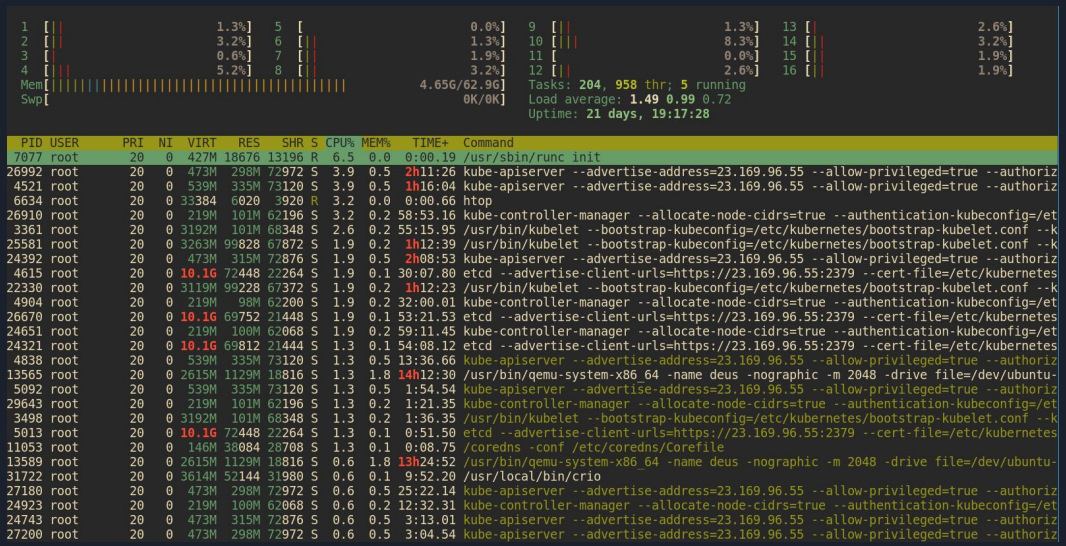
Mosh servers stay around waiting for connections

I know these connections are dead

- `$ pgrep mosh-server | grep -v $PPID | xargs kill`

Kill mosh servers except the current connection

Working with processes: htop



- Aesthetically pleasing
- Interactive (curses)
- Does what all the other tools can do

Part 2:
Init and Services

Two types of processes

Not strict definitions...

Foreground process: chrome, vim, htop.
Started and stopped by the user.

Daemons: background processes like sshd, nginx, postfix, etc.

What is a service?

A service is a **daemon** managed by your init system

Usually a collection of processes working together

How do we control processes

Many ways…

- We can send signals to the process
- We can use CLI tools (apachectl, prosodyctl)
- We can have the init system do the above

What is an init system?

The daemon that manages all other daemons

First process started by the kernel (PID 1)

Starts enough processes to make the system useful

Stops processes on shutdown

Reaps orphans that are zombies

Bare bones init system

Have the user manage the system

Okay, let’s just have init spawn a shell

Bare bones init system

```
pid_t result;
if ( result = fork() > 0 ) {
    /* parent process */
    while ( wait ( NULL ) != result ) { }
} else {
    /* child process */
    execv( “/bin/bash”, NULL );
}
```

Reap all processes

Start other processes

Traditional init systems

Other responsibilities

- Start and stop daemons at certain times (runlevels) during the boot chain
- Launch services
- Send signals to services

How does a traditional init system do it?

There is a script for each service

```
/etc/init.d/$service $action
```

Each script acts on one process

Why are actions better than manually using tools?

Instead of figuring out how to signal every service on my system, I remember 5 actions that apply to *all* services.

If I want details, I can look at the scripts.

What is a “modern” init system

Traditional	**Modern**
Shell scripts that can do *anything*	Declarative configuration files
Stores state in PID files	Stores state in init system
Actions can be customized per script	Consistent actions provided by CLI tool
Difficult to control ordering	Ordering by dependencies or events

A traditional init script - NGINX

```
start() {
    [ -x $nginx ] || exit 5
    [ -f $NGINX_CONF_FILE ] || exit 6
    make_dirs
    echo -n $"Starting $prog: "
    daemon $nginx -c $NGINX_CONF_FILE
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    killproc $prog -QUIT
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    configtest || return $?
    stop
    sleep 1
    start
}

reload() {
    configtest || return $?
    echo -n $"Reloading $prog: "
    killproc $prog -HUP
    retval=$?
```

- Only a subsection of the script
- Tedious to write
- Lots of copy-pasted code
- Keeps state using lock files and PID files - racey!

Modern init system - systemd

Developed at Red Hat

Based off macOS launchd

Heavily parallelized

Dependency and event based

systemd service - NGINX

```
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

Anatomy of a service file

`/etc/systemd/system/my-cool-server.service`

```
[Unit]
Description=My cool server
Requires=network-online.target
After=network-online.target

[Service]
ExecStart=/usr/local/bin/my-cool-server

[Install]
WantedBy=multi-user.target
```

Dependency on network

Run our executable

When service can start automatically

`systemctl` (no arguments)

```
  getty@tty1.service                                loaded active running   Getty on tty1
  haveged.service                                   loaded active running   Entropy daemon using the HAVEGE algorithm
  irqbalance.service                                loaded active running   irqbalance daemon
  keyboard-setup.service                            loaded active exited    Set the console keyboard layout
  kmod-static-nodes.service                         loaded active exited    Create list of required static device nodes
  loadcpufreq.service                               loaded active exited    LSB: Load kernel modules needed to enable c
  lvm2-lvmetad.service                              loaded active running   LVM2 metadata daemon
  lvm2-monitor.service                              loaded active exited    Monitoring of LVM2 mirrors, snapshots etc.
  munin-node.service                                loaded active running   Munin Node
  netfilter-persistent.service                      loaded active exited    netfilter persistent configuration
  networking.service                                loaded active exited    Raise network interfaces
  nginx.service                                     loaded active running   A high performance web server and a reverse
  node_exporter.service                             loaded active running   Prometheus node_exporter
  ntp.service                                       loaded active running   LSB: Start NTP daemon
  postfix.service                                   loaded active exited    Postfix Mail Transport Agent
  postfix@-.service                                 loaded active running   Postfix Mail Transport Agent (instance -)
  quota.service                                     loaded active exited    Initial Check File System Quotas
  rpc-gssd.service                                  loaded active running   RPC security service for NFS client and ser
● rpc-svcgssd.service                               loaded failed failed    RPC security service for NFS server
  rpcbind.service                                   loaded active running   RPC bind portmap service
  rsyslog.service                                   loaded active running   System Logging Service
  serial-getty@ttyS0.service                        loaded active running   Serial Getty on ttyS0
  ssh.service                                       loaded active running   OpenBSD Secure Shell server
  sysstat.service                                   loaded active exited    LSB: Start/stop sysstat's sadc
  systemd-journal-flush.service                     loaded active exited    Flush Journal to Persistent Storage
  systemd-journald.service                          loaded active running   Journal Service
  systemd-logind.service                            loaded active running   Login Service
  systemd-modules-load.service                      loaded active exited    Load Kernel Modules
  systemd-random-seed.service                       loaded active exited    Load/Save Random Seed
  systemd-remount-fs.service                        loaded active exited    Remount Root and Kernel File Systems
  systemd-sysctl.service                            loaded active exited    Apply Kernel Variables
  systemd-tmpfiles-setup-dev.service                loaded active exited    Create Static Device Nodes in /dev
  systemd-tmpfiles-setup.service                    loaded active exited    Create Volatile Files and Directories
  systemd-udev-trigger.service                      loaded active exited    udev Coldplug all Devices
  systemd-udevd.service                             loaded active running   udev Kernel Device Manager
  systemd-update-utmp.service                       loaded active exited    Update UTMP about System Boot/Shutdown
  systemd-user-sessions.service                     loaded active exited    Permit User Sessions
```

systemctl status $service

```
# keur @ supernova in ~ [15:59:50]
$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-10-11 14:26:13 PDT; 5 days ago
     Docs: man:nginx(8)
  Process: 809 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 725 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 837 (nginx)
    Tasks: 9 (limit: 4915)
   Memory: 20.9M
      CPU: 1min 53.810s
```

systemctl start $service

systemctl enable $service

journalctl

Log aggregator built into systemd

View recent logs for a unit

- `$ journalctl -exu $service`

Tail logs (tail -f /var/log/nginx.log)

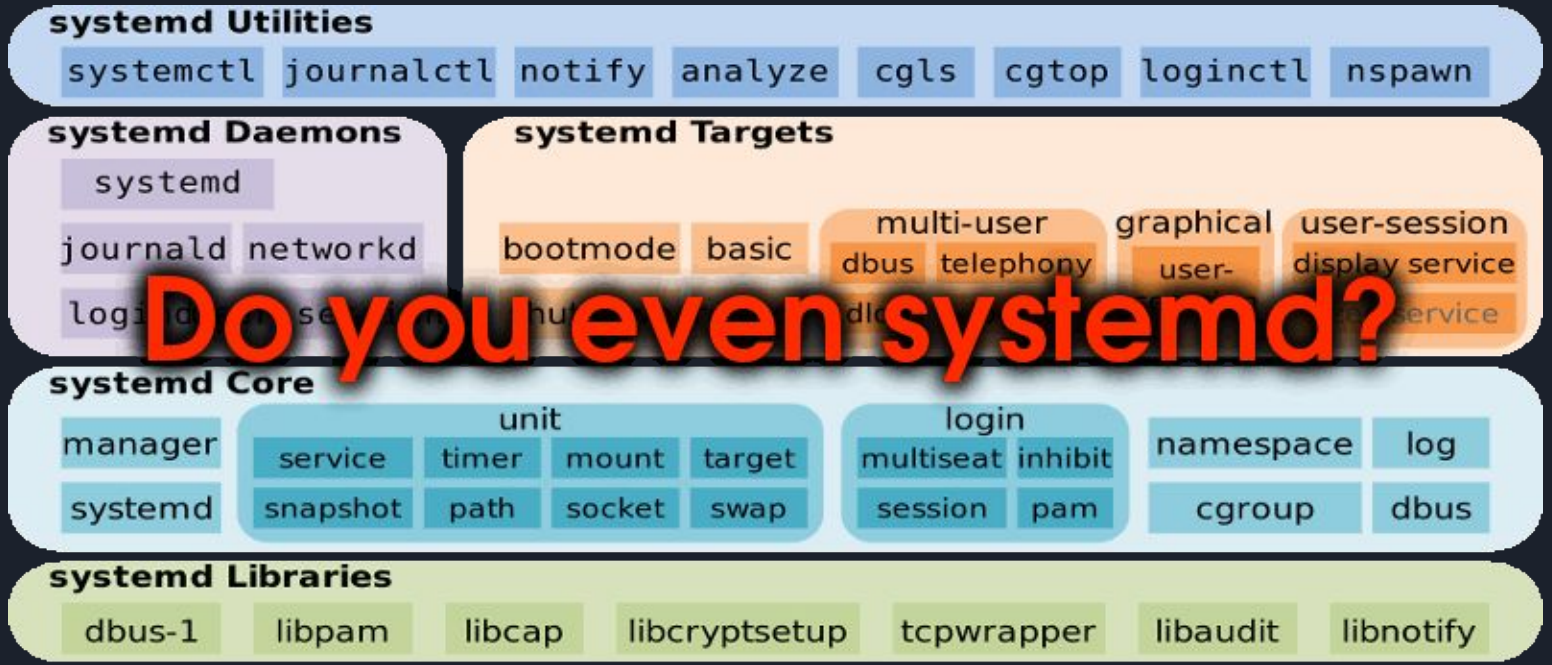
- `$ journalctl -fu $service`

journalctl - Declarative logs

```
$ journalctl --since today --until now
```

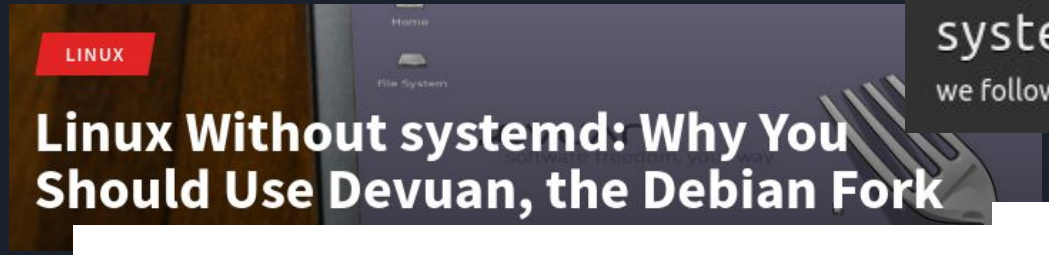
```
# keur @ magneto in /lib/systemd/system [16:11:12]
$ journalctl --since today --until "3 hours ago"  | head -n 15
-- Logs begin at Wed 2018-09-26 21:48:11 PDT, end at Wed 2019-10-16 16:09:05 PDT. --
Oct 16 00:28:42 magneto kernel: Filesystems sync: 0.013 seconds
Oct 16 00:28:42 magneto kernel: Freezing user space processes ... (elapsed 0.007 seconds) done.
Oct 16 00:28:42 magneto kernel: OOM killer disabled.
Oct 16 00:28:42 magneto kernel: Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
Oct 16 00:28:42 magneto kernel: printk: Suspending console(s) (use no_console_suspend to debug)
Oct 16 00:28:42 magneto kernel: wlo1: deauthenticating from c0:8a:de:30:94:dc by local choice (Reason: 3=DEAUTH_LEAVING
Oct 16 00:28:42 magneto kernel: sd 0:0:0:0: [sda] Synchronizing SCSI cache
Oct 16 00:28:42 magneto kernel: sd 0:0:0:0: [sda] Stopping disk
Oct 16 00:28:42 magneto kernel: e1000e: EEE TX LPI TIMER: 00000011
Oct 16 00:28:42 magneto kernel: ACPI: EC: interrupt blocked
Oct 16 00:28:42 magneto kernel: ACPI: Preparing to enter system sleep state S3
Oct 16 00:28:42 magneto kernel: ACPI: EC: event blocked
Oct 16 00:28:42 magneto kernel: ACPI: EC: EC stopped
Oct 16 00:28:42 magneto kernel: PM: Saving platform NVS memory
```

systemd is really complicated



We barely scratched the surface...

systemd is really controversial



systemd-free linux community

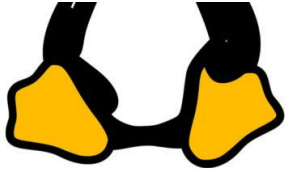
we follow the development of linux away from systemd

systemd is the best example of Suck.

There is a menace which is spreading like a disease throughout the Linux world, it is called systemd.

This is tux

This is tux on systemd



systemd goals

- Performance
- Simple UX
- Providing “building blocks” for an operating system

Ended up re-implementing many existing tools along the way...

My take...

systemd is easy to use for simple things

Fast init systems are actually pretty complicated

At this point, you don’t really have a choice anyway...

its not dns